Carnegie Mellon
Software Engineering Institute

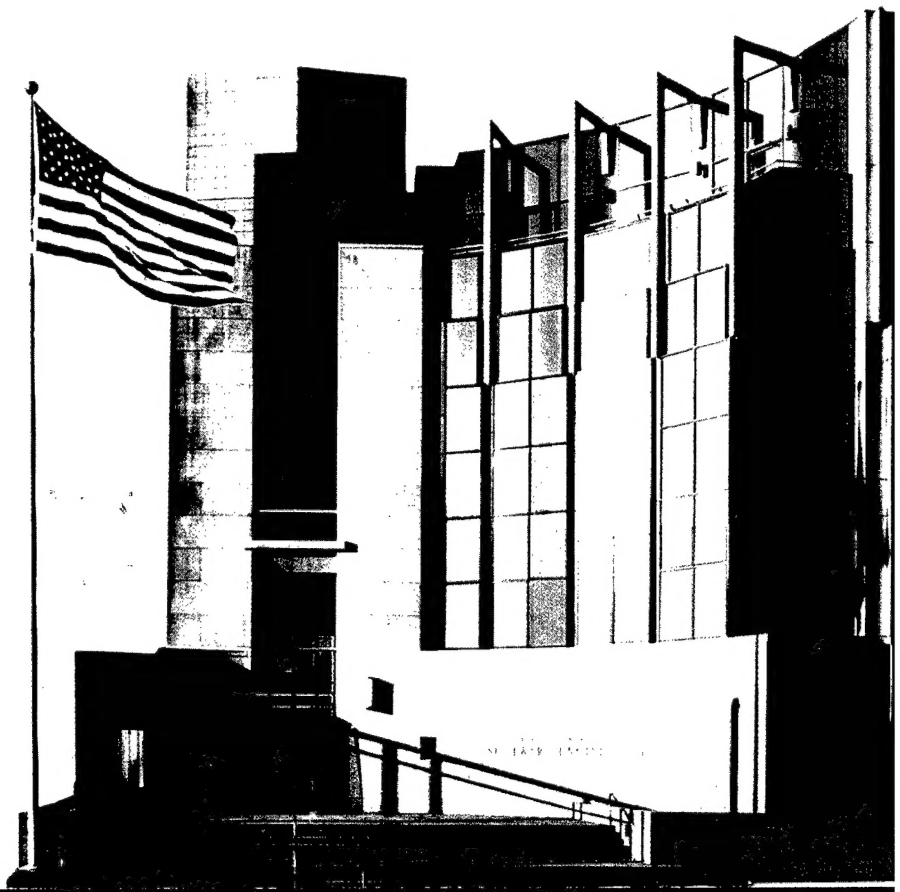
SACAM: The Software Architecture Comparison Analysis Method

Christoph Stoermer
Felix Bachmann
Chris Verhoef

December 2003

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

TECHNICAL REPORT
CMU/SEI-2003-TR-006
ESC-TR-2003-006



Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

SACAM: The Software Architecture Comparison Analysis Method

CMU/SEI-2003-TR-006
ESC-TR-2003-006

Christoph Stoermer
Felix Bachmann
Chris Verhoef

December 2003

Architecture Tradeoff Analysis Initiative

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Christos Scondras
Chief of Programs, XPK

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Executive Summary

The Software Architecture Comparison Analysis Method (SACAM) provides organizations with a rationale for an architecture selection process by comparing the fitness of software architecture candidates being used in envisioned systems. The method addresses organizations that select software architectures to

- **explore** the support of existing software product architectures for required software product lines
- **decide** among competitive software architectures proposed by subcontractors

The SACAM contributes to the selection process by providing results from software architecture analysis. Software architectures are critical for achieving an organization's business goals and provide a more manageable level of abstraction for handling information than the code of the software.

The comparison is performed in a series of steps. After a preparation step, the comparison criteria that serve as a yardstick in the method are collated and then refined into concrete quality attribute scenarios. The scenarios and existing architectural documentation of the architecture candidates are used to identify the relevant information for the comparison. This information is extracted from the architectural documentation and analyzed to determine how well the required scenarios are supported. The stakeholders score each architecture on a scenario basis, which leads to a recommendation for selection. The scores might reflect weights that are provided by the stakeholders for the criteria. The artifacts generated during the course of the method can be used for subsequent processes, such as an architectural commonality and variability analysis for a software product line migration.

Based on the current status of the SACAM, it is our belief that using this method comes with three major benefits. First, it offers acquiring organizations a qualitative approach for making selections among alternative architectures beyond quantitative approaches, such as cost. Intuition is replaced by an analysis framework. Second, the criteria approach is goal-oriented. For example, architectural commonalities and differences are condensed and interpreted with regard to the criteria qualities. Third, comparison on a manageable architectural level offers the ability to evaluate changing business goals in the requirements space with solutions in the design space.

Abstract

Comparing software architectures for any nontrivial system is a difficult task. Software architectures are designed with particular requirements and constraints, and are often poorly documented. However, organizations often need to select a software architecture for future development from several candidate architectures. The Software Architecture Comparison Analysis Method (SACAM) was created to provide the rationale for an architecture selection process by comparing the fitness of architecture candidates for required systems. The SACAM compares architectures based on a set of criteria derived from the business goals of an organization. The SACAM was developed in a technical reuse context where an organization investigated architectural commonalities and differences to explore architectural designs for a software product line architecture. This report outlines a first version of the method and its underlying concepts.

1 Introduction

This report describes the Software Architecture Comparison Method (SACAM). A software architecture comparison is necessary when organizations have to select software architectures from a set of candidate architectures. The SACAM was developed in a technical reuse context to help organizations explore architectural designs for use in new software product line architectures. However, the SACAM may well be suitable in other application areas, such as selecting among software architectures proposed by several contractors.

The goal of SACAM is to provide a rationale for an architecture selection process by comparing the fitness of architecture candidates for envisioned systems.

This goal is achieved by providing solutions for two major objectives:

1. the extraction of comparable architectural views of each candidate at similar levels of abstraction
2. criteria collation and analysis of the candidate architectures

The SACAM uses several architecture techniques developed by the Software Engineering Institute (SEI[SM]) to compare the architecture candidates. These techniques are outlined in Section 2.

The SACAM compares the architectures of software systems and not the implementation code. The major reasons are

1. **business goals**. Software architectures determine how well an organization can achieve its business goals. An architecture may have been built to enable or limit the addition of features demanded by customers, to allow or hinder modifications, or to promote or restrict component reuse. Consequently, information about a software architecture provides an important basis for deciding which software is better suited for products supporting the organization's future.

2. **the level of abstraction**. Software can be compared on several different levels. It is possible to compare the requirements, but that does not address how well the software actually realizes the requirements. On the other hand, at the implementation

[SM] SEI is a service mark of Carnegie Mellon University.

level, it is clear how well requirements are fulfilled, but comparing different software is almost impossible because of the huge amount of information. Comparing software architectures provides a manageable level of information. Architecture reconstruction techniques can be used to ensure that the implemented software conforms to the documented architecture and may be used to improve existing documentation.

Our experience is currently limited to the comparison of two architecture candidates. However, more candidates are possible from the perspective of the method. The organization's business goals describe global requirements of an envisioned system and therefore build the basis for the comparison criteria. The criteria provide the yardstick for the comparison. The better an architecture fulfills the criteria, the more promising it is for the envisioned system.

The SACAM consists of the following steps:

1. **Preparation**
   Identifies the relevant business goals needed in the comparison and examines available documentation for each architecture candidate.
2. **Criteria Collation**
   Derives comparison criteria from the business goals and refines them to quality attribute scenarios.
3. **Determination of Extraction Directives**
   Determines the architectural views, tactics, styles, and patterns that are looked for during the following extractions to find supporting evidence for the scenarios of Step 2.
4. **View and Indicator Extraction**
   Extracts the architectural views for each candidate according to the extraction directives from step 3. Detects indicators that support the quality attribute scenarios from Step 2. Architecture recovery techniques may be needed to generate relevant views.
5. **Scoring**
   Scores the fitness of a candidate architecture to support the criteria.
6. **Summary**
   Summarizes the analysis results and provides a recommendation for the decision-making process.

The outputs of the method include the scores and related reasoning for each candidate, along with the generated artifacts, such as architectural documentation.

The examples in this report are extracted from a context where an organization wants to streamline the software for two existing, somewhat similar, products into a new software product line. The organization wants to determine which of the existing software architectures is best suited to form the basis for the new product line. The first comparison candidate is a garage door opener. The second candidate is a sunroof system in an automobile. Both products have similar features, such as opening and closing a door or sunroof. However, the

systems are from different domains. The software of the garage door opener is part of a home integration system, whereas the sunroof is part of an automotive body electronics system.

Section 2 of this report relates the SACAM to other similar and foundational techniques developed by the SEI. An overview of the essential steps of the method is given in Section 3. Section 4 explains the method's concepts, terminology, and analysis framework to map the criteria scenarios onto the candidate architectures. Section 5 describes the SACAM steps using examples from the sunroof/garage door system. Finally, conclusions and directions for future work are presented in Section 6.

2 Related Work

The SEI has developed proven methods and introduced techniques that form the basis for the SACAM. We will explain these techniques and their relation to the SACAM in this chapter.

To extract comparable information from software architectures and perform the collation and analysis of the comparison criteria, the SACAM uses SEI techniques as illustrated in Figure 1. In the center, Figure 1 shows the SACAM method with its objectives: *View Extraction*, *Criteria Collation,* and *Analysis*. *View Extraction* is achieved by using documentation standards and reconstruction techniques. In Figure 1, almost no architectural documentation is available for Software B. *Criteria Collation* is achieved with scenario generation techniques, and *Analysis* is done through architectural tactics and metrics techniques.

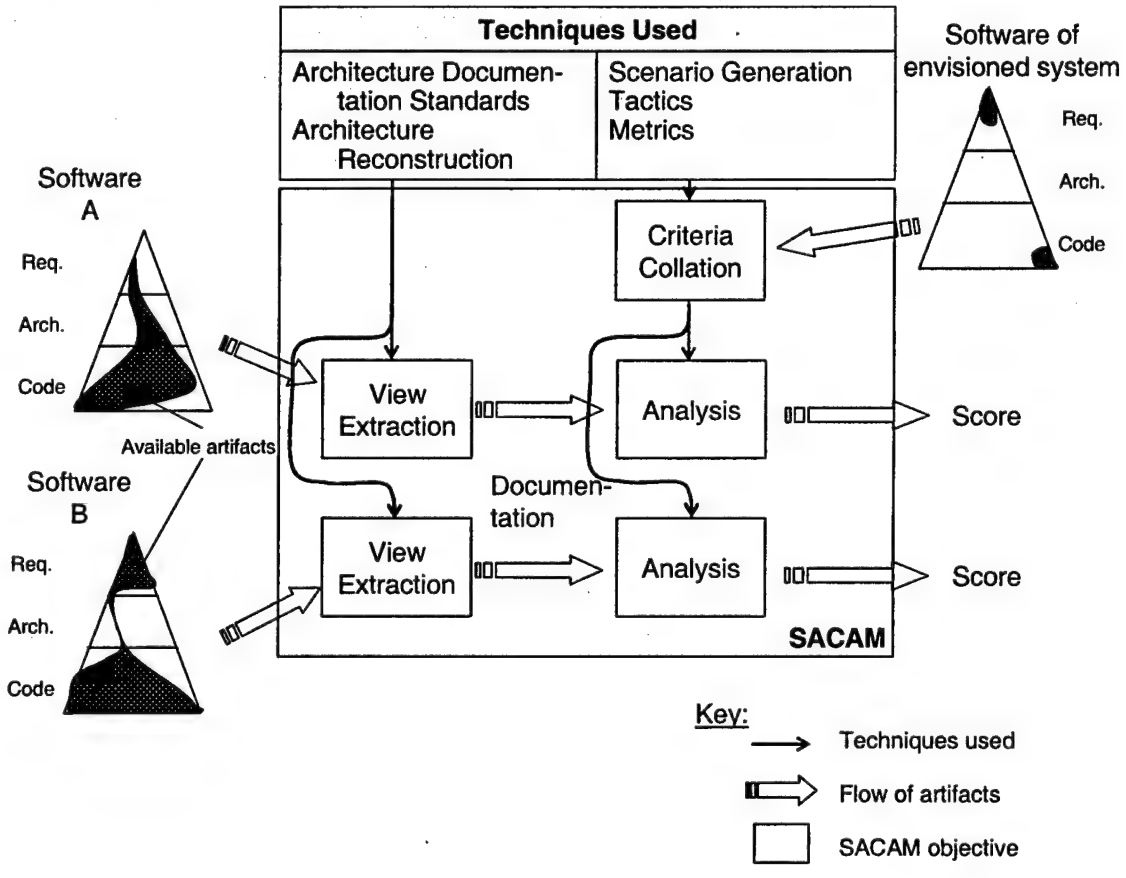


Figure 1: Architecture Techniques Used by the SACAM

Each technique contributes to the comparison of software architectures. These techniques are used to generate the necessary artifacts that are then analyzed to provide the final scores for each architecture.

- **Scenario Generation**
  The SACAM requires criteria that are derived from the business goals of an organization for an envisioned software system. The criteria are articulated in quality attributes that are further refined into quality attribute scenarios. *Scenario generation* is a technique for capturing quality attributes and refining them into quality attribute scenarios. SEI methods that incorporate scenario generations are the Architecture Tradeoff Analysis Method[SM] (ATAM[SM]) [Clements 02b] and the Quality Attribute Workshop (QAW) [Barbacci 03].

- **Tactics**
  To achieve particular qualities that are addressed with scenarios, developers structure the software in particular ways. Bachmann and associates, introduce the notion of *tactics*—strategies to achieve quality attribute goals [Bachmann 03]. The SACAM uses tactics in the analysis as indicators to evaluate if the extracted views support a criterion articulated as a quality attribute scenario. Collections of tactics are available for a variety of quality attributes [Bass 03].

- **Metrics**
  Metrics support quantitative analysis that provides useful indicators of overall complexity and locations where change might be difficult or most likely. Metrics are used in the SACAM on the code level, if available, or on a detailed design level. For example, metric sets refer to Arora and associates [Arora 95] and Faust and Verhoef [Faust 03].

- **Architectural Documentation Standards**
  The SACAM requires the availability of architectural documentation to perform the comparison criteria analysis. Experience shows that architectural documentation across systems is heterogeneous. For example, there are differences in notations, stakeholders, level of documentation detail, and scope. One of the SACAM's challenges is to obtain comparable architectural documentation to prevent the comparison of apples and oranges. The SACAM uses the "views and beyond" architectural documentation approach as provided by Clements and associates [Clements 02a].

- **Architecture Reconstruction**
  The architectural documentation used to perform the comparison might be unavailable, insufficient, or out of date. In these cases, the architecture has to be reconstructed. However, not all architectural views have to be reconstructed, only the relevant views. For example, if the intention is to find the architecture that is best suited to support modifiability, views showing module dependencies are more important than process views. This goal-oriented approach is used in the Quality-Attribute-Driven Software Architecture Reconstruction (QADSAR) method [Stoermer 03].

SM Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

3 Method Overview

An overview of the SACAM is shown in Figure 2. The inputs to the method include:

- **architecture candidates** – the architectures that should be compared. An important consideration is the quality of the architectural documentation along with the availability of architecture expertise. Note that the architectures don't have to be implemented.
- **business goals** – the source of the comparison criteria. These criteria provide the yardstick for the comparison and are derived mainly from the business goals.

The inputs are important factors for a successful application of SACAM. Therefore, step 1 of SACAM ***(Preparation)*** examines the available inputs to prepare a successful application of the method.

In the course of step 2 ***(Criteria Collation)***, a set of criteria for the architecture comparison is identified. A criterion formulates a requirement for the architecture to support the organization's business goals. For example, having a highly modifiable architecture (criterion) may help to achieve fast time-to-market requirements (business goal). Criteria are refined into quality attribute scenarios.

Step 3 ***(Determination of Extraction Directives)*** determines directives for the architectural information that needs to be taken into account during the extraction process. The directives assist the extraction by providing a list of common viewtypes, suggested tactics, styles, patterns, and metrics that provide information on what to look for in the architectural documentation of each candidate.

Step 4 ***(View and Indicator Extraction)*** extracts the required views from the available architectural documentation for each candidate architecture according to the extraction directives of Step 3. The views comprise architectural elements and their relations. In addition, the extraction elicits indicators that provide evidence for the support or hindrance of particular quality attributes. Indicators are either architectural information (such as tactics or patterns) or low-level metric information. Metric information is extracted from implementations (if available) or detailed design artifacts. The indicators provide evidence for the scoring in Step 5. For example, if the indicator extraction determined the tactic "intermediary" in a modifiability scenario for a protocol layer, and a complexity metric shows low values for the affected modules, then it is very likely that the architecture supports modifiability for the protocol layer.

In Step 5 ***(Scoring)***, each criterion is scored for an architecture candidate. The scoring is based on the evidence provided by Step 4, and the quality attribute scenarios determined during step 2. The scoring might consider weights that are provided by stakeholders for the criteria. The scoring provides the reasoning and the resulting score for how well the criteria scenarios are supported by a candidate.

Step 6 ***(Summary)*** summarizes the results of the analysis.

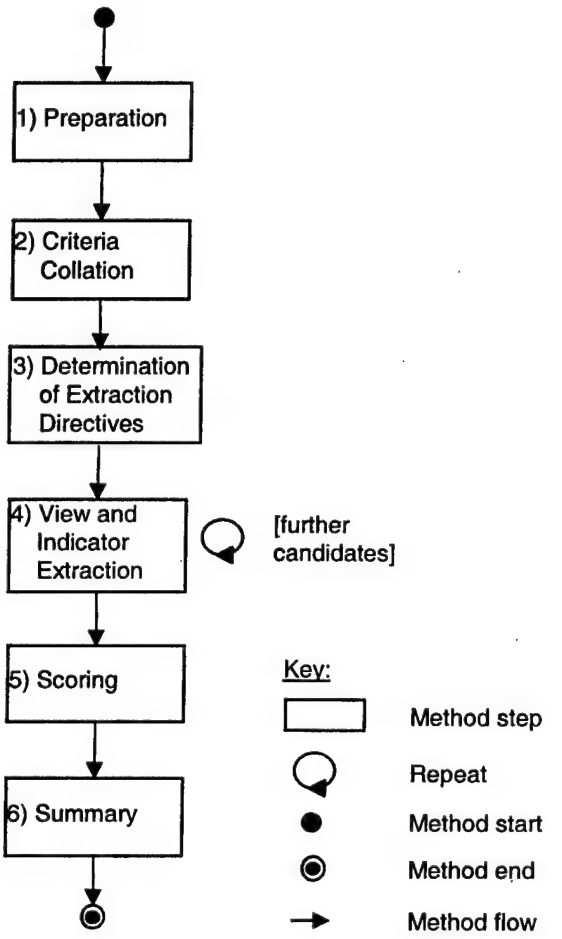


Figure 2: SACAM Overview

The outputs of the method include a recommendation for the decision-making process, the scores and the related reasoning for each candidate, and the generated artifacts, such as architectural views, tactics, and scenarios. The artifacts generated during the course of the SACAM are useful for subsequent processes, such as commonality and variability analysis in product line architecture designs, or the exploration of cross-functionality in multiple architectures.

4 Concepts and Terminology

In this section we describe the underlying concepts and terms used by the SACAM. We will address the two major objectives:

1. extraction of comparable architectural views of each candidate at similar levels of abstraction
2. collation and analysis of criteria that provide a comparison yardstick

Finally, we will explain the artifacts that are created during the application of the SACAM.

4.1 Heterogeneity

Comparing software architectures is typically done in heterogeneous contexts. For example, consider the architectures of the garage door opener and the sunroof. Both systems are developed in different departments within an organization. The documentation for the sunroof consists of a functional requirements specification and a detailed design description. The documentation for the garage door opener comprises a 20-page architecture document. The SACAM has to meet the challenge of differences in level of abstraction and form. A closer look at the documents reveals differences with respect to terminology, concepts, notations, levels of detail, and views. A successful comparison requires comparable architectural documentation.

4.2 Views

Confronted with heterogeneous architectural documentation, the SACAM has to identify and extract architecture representations that allow comparisons. The first part in this task is an evaluation of the architectural documentation to determine its usefulness for the criteria analysis. Some architectural information may not be relevant. For example, if the goal is to find architecture that supports several user interfaces for the garage door and the sunroof, then views showing module dependencies are more important than process views. Consequently, the evaluation and identification of relevant architectural information is driven by the comparison criteria.

Depending on the evaluation results, it might be necessary to use architecture reconstruction techniques [Krikhaar 99, Kazman 97] to derive more suitable views from implementation

artifacts, such as code or build files. A *view* is a representation of a set of system elements and relationships among them [Clements 02a].

After the evaluation and identification of relevant architectural documentation, heterogeneous documentation has to be *normalized*, that is, converted to views with defined notations and comparable levels of abstractions. We provide an example in Section 4.5.2.

4.3 Criteria

Comparing software architectures implies a set of criteria. A comparison without any criteria produces no sound reasoning about a selection. For example, if the business goal is to increase reusability among different product architectures, a specification of future commonly used artifacts is required. The criteria that can be used to measure the candidate architectures by how well they support the business goals include

- distribution of functionalities included in future commonly used artifacts throughout the architecture
- degree of dependency of those functionalities on other, product-specific functionalities

Criteria provide the yardstick in the comparison, so criteria must be analyzed for each candidate.

Comparison criteria are foremost quality driven, for example

- Acquirers need not only quantitative aspects but also qualitative aspects for a down select after a competitive solicitation.
- Marketing departments ask for interoperability with third-party software.
- Technical management demands the localization of market-specific adaptations in the software.
- Technical support asks for improvements to maintenance capabilities in the software.

Criteria represent requirements that are important for some stakeholders to achieve their business goals. The collation of criteria is achieved in the SACAM by using scenario generation techniques as proposed by the ATAM [Clements 02b] or the QAW [Barbacci 03].

4.4 Indicators

Indicators are information extracted from the architecture that assist in determining how well an architecture fulfills one or more comparison criteria. For example, if one criterion is to prevent unauthorized users from using a system, the architecture that already implements security concepts is better suited for future use than one without security concepts. The infor-

mation "implements security concepts" is then used as an indicator for the comparison. On the other hand, if security is not a criterion, this information would be irrelevant.

The SACAM uses indicators to provide evidence for the architecture scoring with respect to the criteria.

Indicators can be extracted from many available software artifacts. The artifacts and techniques that can be used to extract indicators include

- architecture – the use of architectural styles [Klein 99] or more detailed architecture tactics [Bachmann 03]
- detailed design – the use of design patterns [Gamma 95]
- code – metrics, such as complexity, function points, fan in/fan out, and so forth

Metrics support quantitative analysis that provides useful indicators of overall complexity and locations where change might be difficult. For metric set examples, see the work of Arora and associates [Arora 95] and Faust and Verhoef [Faust 03]. Some metrics that can be used in the comparison include

- number of function points for architecture design elements
- number of events (synchronous/asynchronous) to which an architecture design element has to react
- complexity values for architecture design elements [McCabe 70]
- average execution time for architecture design elements
- volatility, utility, and specificity of a code unit [Faust 03]

However, it is important to collect the right data—not just any data. A goal-oriented approach is presented in the Goal/Question/Metric (GQM) paradigm. A rich source for GQM is provided by Basili [Basili 92].

No matter which artifact is used, extracted indicators provide evidence about quality attributes.

4.5 The Analysis Framework

At the heart of the SACAM is the analysis framework. It offers a reasoning framework for why an architecture does or does not comply with the comparison criteria.

The analysis framework distinguishes between the analysis of functional-driven and quality-driven criteria (see Figure 3). Both are formulated in the quality attribute scenarios that are

generated from the criteria. The scenarios have to be mapped onto the extracted architectural views of the candidate architectures. This requires clear identification of the design elements that exist to satisfy a scenario. The term "design element" is used as a placeholder for various architecture artifacts, such as process, subsystem, module, or component. The particular meaning is defined by the context of the corresponding architectural view [Clements 02a].

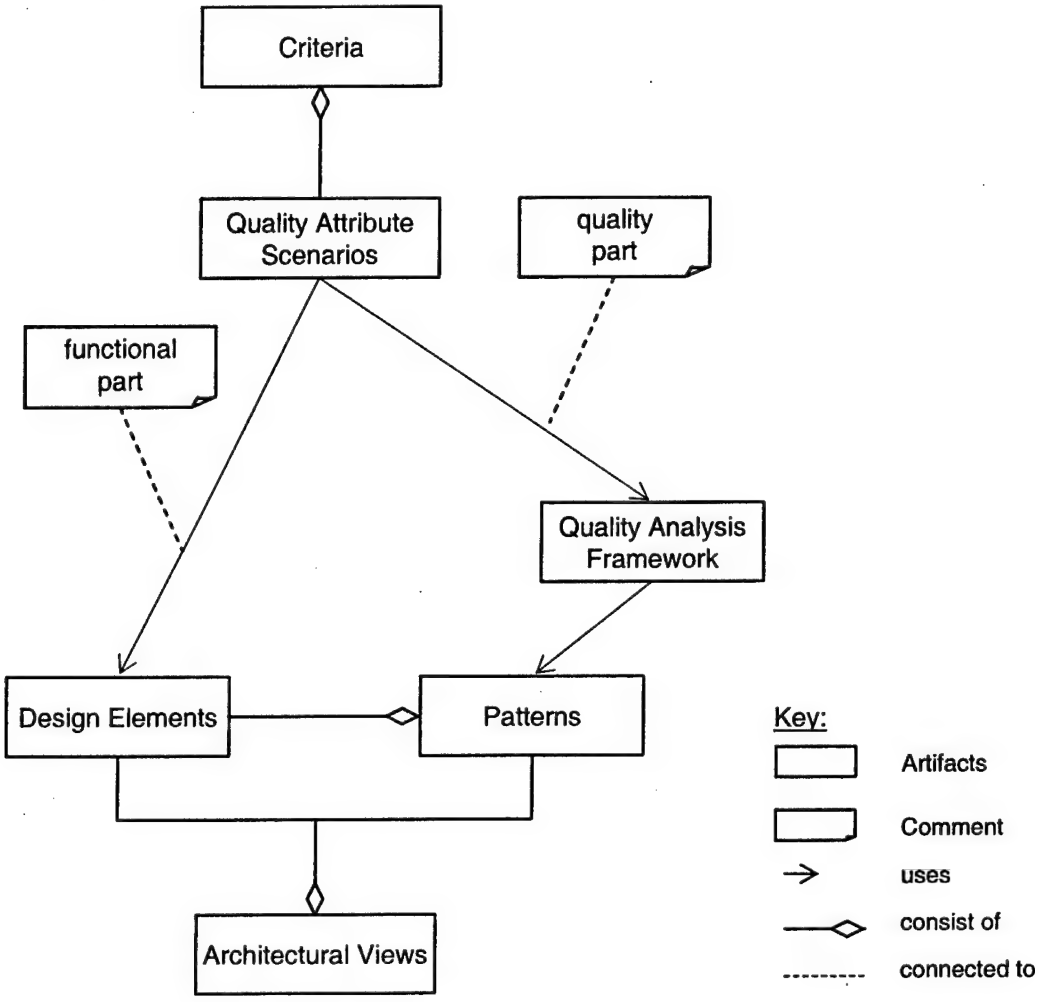


Figure 3: Analysis Framework

The functional part of a *quality attribute scenario* usually maps to the behavior and responsibilities of *design elements*, while the quality part usually maps onto the structure expressed in the form of patterns. The term *pattern* is used here in a generic way. Examples are design patterns [Gamma 95], architecture patterns [Buschmann 96], architectural styles [Klein 99], or just conventions, such as naming conventions. The common characteristic is a recurring, identifiable element in the architecture. Patterns influence the achievement of quality goals. They contribute, for example, to the achievement of performance or modifiability goals. More and more publications address with this topic [Noble 01, Chung 00]. We revisit the important relation of quality attribute goals and patterns in subsequent sections.

4.5.1 Quality Attribute Scenarios

Quality attribute scenarios capture criteria that are driven by the achievement of quality attributes. We use "quality attribute scenarios" according to the following definition: "A quality attribute scenario is a quality attribute requirement of a system. It consists primarily of a stimulus and a response. The stimulus is a condition that needs to be considered when it arrives at a system and the response is the (measurable) activity undertaken after the arrival of the stimulus. In addition to the stimulus and response, a quality attribute scenario includes the source of the stimulus, the environment under which the stimulus occurs, the artifact that is stimulated, and how the response is to be measured" [Bass 03]. For example, in the scenario: "*The sunroof must be able to accept and execute user commands 250 ms after power-on*" the stimulus is *power-on,* the response is *accept and execute user commands,* and the response measure is *250 ms.* The scenario does not deal explicitly with the source of the stimulus, environment, and artifact. The scenario description might be extended if more detailed information is required.

We also use the concept of general scenarios as described by Bachmann and associates [Bachmann 03]. General scenarios describe quality attribute requirements in a system-independent manner. System-specific quality attribute scenarios are called concrete quality attribute scenarios. The combination of concrete and general scenarios suggests quality attribute models to use for analysis, and provides input for those models.

4.5.2 Quality Analysis Framework

In this section, we show how the different artifacts, produced during an application of the SACAM, are related. The quality analysis framework provides

- assistance in searching for particular indicators in the architectural documentation
- reasoning as to whether quality attribute scenarios are satisfied by an architecture candidate

To achieve particular qualities described by scenarios, developers decide to structure the software in a particular way. For example, the sunroof start-up scenario from the previous section is a time performance scenario that can be achieved with a set of particular tactics, such as "reduce computational overhead" or "manage event rate" [Bass 03]. A quality attribute framework provides a set of potential tactics that helps achieve a particular quality attribute. A collection of tactics for a variety of quality attributes, such as modifiability, performance, usability, testability, and availability, are presented in the book *Software Architecture in Practice* [Bass 03]. The set of tactics for a particular quality attribute is used in the SACAM to assist the extraction of views from the architectural documentation by providing information on what to look for in the documentation.

On a more detailed design level, patterns also support particular quality attributes. For example, "fixed allocation," "pooled allocation," or "sharing," are more time performance oriented than "reference counting" or "paging." Noble and Weir provide a collection of patterns for achieving particular quality attribute goals [Noble 01]. On the even more detailed code level, metrics can be used to extract complementary indications about quality attribute support.

In summary, the SACAM uses a quality analysis framework in a goal-oriented way. Therefore, criteria collation using the method must occur before the extraction of views.

The quality analysis framework assists the extraction of indicators by providing extraction directives. These directives provide a list of common viewtypes and suggested tactics, styles, patterns, and metrics that indicate what to look for in the architectural documentation of each candidate.

Further, the quality analysis framework provides the reasoning for the criteria scoring. For example, an organization knows that the exchange of the communication protocol is important for the success of the envisioned system. To understand how well the existing systems support that exchange, the following criterion is defined:

Criterion	Exchange of communication protocol The organization wants to exchange an externally provided communication protocol for an envisioned sunroof/garage door software product line.

To be more precise, the criterion is refined into a concrete scenario using the general scenario generation table for modifiability [Bass 03, p. 83]. The table provides information that should be part of the scenario.

Concrete Scenario	The application developer exchanges the Controller Area Network (CAN) communication protocol of the sunroof/garage door software with a FlexRay or similar protocol within one person-day during development time.

The concrete scenario provides values for the quality attribute reasoning and the expected response for an architecture candidate (one person-day of effort). For example, the scenario states that a part of the system has to be replaced. Therefore, we would look for the use of the "localization" tactic. We also know from the scenario that the change must be made during development time, not runtime. Therefore, we would expect the module viewtypes to be used during development of the software and not runtime views. We now examine the architectural documentation of the sunroof system.

Architecture Candidate	Sunroof System

The quality attribute framework for modifiability is concerned with the allocation of responsibilities to modules and their dependencies. The required view should show modules, allocated responsibilities, and dependencies. Consequently, we should examine the "uses view" of the sunroof system.

Views	Uses View

A uses view was not available for the system. The software was reconstructed using the Quality-Attribute-Driven Software Architecture Reconstruction (QADSAR) method [Stoermer 03]. The major source was therefore

Source	The source code The reconstructed uses view

The source and the quality attribute model provide the basis for the indicator extraction. The set of suggested tactics is taken from the modifiability model.

Indicators — Metric indicator

- number of modules containing communication protocol dependencies
- number and strength of the dependencies of these modules on others. The dependencies should comprise functional and data access dependencies.

Patterns: No patterns detectable
Tactics: Semantic coherence of the protocol software

The sunroof system does not provide tactics such as "abstract common services" or "generalize module." The tactic "semantic coherence" is suggested because the protocol software is packaged as a single software piece. There are no tactics for preventing ripple effects. The number of dependencies provided by the metrics indicates significantly more time to exchange the communication protocol than the required one-day effort.

Score	2

The scoring is done between 0 – 10, where 0 means that no support is provided, and 10 completely satisfies the criterion. The resulting response (significantly more than one person-day of effort) with the low score of "2" provides the feedback for the criterion.

5 The Method

In this section, we present the comparison method based on the concepts in Section 4 and use the sunroof/garage door example to illustrate the application of the method.

The SACAM consists of the following steps:

1. Preparation
2. Criteria Collation
3. Determination of Extraction Directives
4. View and Indicator Extraction
5. Scoring
6. Summary

We describe the method steps by using a template that contains the following parts:

Brief Description – a short description of the step's purpose

Input – the expected inputs for the step

Activities – a description of the activities that have to be performed during the step

Output – the outputs of the step

Dynamics – the participants involved and the execution form of the step

Example – extracted from the sunroof/garage door application

Step 1 – Preparation

Brief Description: Lays the foundation for the application of the method. The application context, business goals for the envisioned system, comparison candidates, scope, and stakeholders are identified. The available architectural documentation is examined and a plan for the SACAM steps is developed.

Input:

- business goals of the envisioned system
- set of comparison candidates

Activities:

- **Identify the application context**. The SACAM can be performed in a variety of application contexts. For example, in a product line migration context, the architectures of candidate systems are explored to gain qualitative information as to how well the different architectures would support the envisioned software product line. Further application contexts are possible, such as the selection among competitive architectures proposed by several contractors for an envisioned software system.
- **Identify the comparison stakeholders**. The comparison stakeholders promote the comparison criteria, perform the scoring and verify the method results.
- **Select the comparison candidates**. The number of comparison candidates should be limited to control the total amount of effort required.
- **Identify the candidate stakeholders**. The candidate stakeholders represent the candidates, provide the necessary source information for each candidate, and are available for interviews. Comparison stakeholders and candidate stakeholders might be the same person or group of people.
- **Present the SACAM**. At the beginning, the method is presented to the organization by the SACAM team. The goal of the presentation is to develop common expectations about the results of the method and the activities necessary to achieve these results.
- **Identify the business goals for the envisioned system**. The business goals that should be supported by the envisioned system, or systems, are elicited from the stakeholders.
- **Present the candidates.** The selected candidates should be presented by the software architects (if they are still available). This includes goals for the architecture, important design decisions, and major components of the architecture.
- **Examine the available architectural documentation.** This activity determines how detailed and current the available documentation is.
- **Outline the SACAM plan.** The plan contains the SACAM steps with the activities (as outlined in this chapter), the participants, and the schedule. The plan might be reviewed during the course of the method, for example if architecture reconstructions are necessary.

Outputs:

- selected comparison candidates for SACAM
- SACAM plan

Dynamics:
The preparation step should provide buy-in for the stakeholders. This can be done in a preparation meeting and a half-day workshop. The meeting includes the first four activities. The workshop starts with the presentation of the SACAM and ends with the outline of the SACAM plan.

Example:
The application context of the example is an organization that is a market leader for garage door openers, sunroofs, and other electronic products. The organization investigates various cost consolidation approaches. One approach is to streamline the software for sunroofs and garage door openers in a software product line.

The company is well known for high quality and incorporating the latest technologies. Its primary business goal is to sell about 100,000 garage door openers and sunroofs a year, including variants for low-end, mid-range, and high-end markets.

Table 1 illustrates a generalized version of the SACAM plan for the sunroof/garage door opener example.

Table 1: SACAM Plan

Step		Participants	Time/Format
2	Criteria Collation	Comparison stakeholders, candidate stakeholders	1-day workshop
3	Determination of Extraction Directives	SACAM team	1 week, offline
4	View and Indicator Extraction for sunroof software	SACAM team, candidate stakeholders	½-day workshop with sunroof architects
	View and Indicator Extraction for garage door software	SACAM team, candidate stakeholders	½-day workshop with garage door architects
5	Scoring of both candidate architectures	SACAM team, comparison and candidate stakeholders	Full-day workshop
6	Presentation of results (end of Scoring Workshop)	SACAM team, comparison and candidate stakeholders	45 min. at end of full-day workshop of Step 5, including discussions
	Report	SACAM team	3 days, offline

Step 2 – Criteria Collation

Brief Description: The business goals are used to create a set of comparison criteria that serve as the yardstick for the comparison. The criteria are refined into quality attribute scenarios. Criteria might be prioritized by the comparison stakeholders.

Input:

- the business goals for the envisioned system

Activities:

- **Identify criteria.** Criteria describe desired properties of the envisioned system. They can be collected in a brainstorming session with a consolidation that follows.
- **Prioritize criteria.** The stakeholders should prioritize the criteria. Prioritization can be done by voting or by finding a consensus by discussing each criterion and determining its weight.
- **Refine criteria.** Criteria are refined into concrete quality attribute scenarios according to the description in Section 4.5.1. Note that a criterion can comprise several scenarios.

The result of the activities is a collection of prioritized criteria described as quality attribute scenarios.

The ATAM [Clements 02b] and the QAW [Barbacci 03] offer a rich source of examples and experiences with scenario elicitation.

Output:

- a list of prioritized criteria refined as quality attribute scenarios

Dynamics:

Identifying, refining, and prioritizing the criteria is typically done during a one-half- to one-day criteria collation workshop. This workshop requires the participation of the comparison stakeholders and can be performed immediately after the workshop of Step 1.

Example:

The elicited quality attribute scenarios for the envisioned software product line include

#	Scenario	Quality Attributes	Criteria Priority
1	If an obstacle (person or object) is detected by the system during descent (garage door) or closing (sunroof), it must halt (alternatively reopen) within the legal ranges given by standards of the particular market (currently <= 0.02 sec.).	Safety, Performance	High
2	The processor used in different products will differ. The product-specific architecture for each product should be derivable from the product line architecture with minimum effort.	Variability	High
3	The envisioned system should be accessible for diagnosis and administration from a network using a product-specific diagnosis protocol.	Modifiability	Medium

Step 3 – Determination of Extraction Directives

Brief Description: Determines the type of views that should be extracted from each candidate. Generates a list of architectural tactics that would, if identified in the extracted views, provide evidence for or against the quality attribute scenarios from Step 2.

Input:

- quality attribute scenarios from Step 2
- architectural documentation for each candidate

Activities:

- **Determine a set of viewtypes that is common across all candidates.** The architectural documentation must include the necessary information. For sample viewtypes, see the work of Clements and associates [Clements 02a]. Viewtypes include
  - Module Viewtype – documents the units of implementation, especially if modifiability is important
  - Component and Connector Viewtype – documents runtime units with their concurrency behavior, especially if qualities like performance, availability, or security are important
  - Allocation Viewtype – documents the relationship between the software and the development and execution environments

- **Gather suggested architectural tactics, styles, patterns, and metrics**. There are collections of architectural tactics available to support the achievement of particular quality attributes [Bass 03]. Architectural styles and patterns also support or hinder particular quality attributes. These suggested tactics, styles, and patterns assist the following extractions by providing information about what to look for in the architectural documentation.

Output:

- a list of required viewtypes and what type of information they should contain; the list also contains references to the quality attribute scenarios addressed by the viewtypes
- a list of suggested tactics, styles, patterns, and metrics that should be searched for during the view and indicator extraction step

Dynamics:

Step 3 typically requires time for the analysis and does not follow immediately after the workshops of steps 1 and 2. Step 3 is performed by the comparison team.

Example:

Table 2 shows the extraction directives for scenario 2. In addition, the following metrics should be extracted:

- number of modules containing processor dependencies
- number of dependencies of these modules on others

Table 2: List of Extraction Directives for Scenario 2

View	Tactics	Patterns
Uses View, showing modules with processor dependencies	*Anticipated Variants*—to localize functionality that is dependent on the variants (different processors) *Generalized Variation Points*—a defined way to access any possible variant without adaptation *Limited Possible Variants*—make the definition of a generalized variation point easier (e.g., only processors from manufacturer X with support of standard C language) *Intermediary Used*—break undesirable dependencies to the variants (e.g., hide access to processor hardware such as I/O)	Layered pattern for: *Generalize Variation Points*—access to a layer only via an interface *Intermediary Used*—a layer can act as an intermediary between two adjacent layers The *Anticipate Variants* tactic has no pattern because it is a task of the designer The *Limit Possible Variants* tactic also has no pattern because it involves a business decision of the organization

Step 4 – View and Indicator Extraction

Brief Description: Extracts the required architectural documentation, which is then examined for used tactics, styles, and patterns as described in Step 3

Input:

- extraction directives from Step 3
- architectural documentation for each candidate
- implementation code, if necessary

Activities:

- **Extract views from the architectural documentation.** The selected viewtypes from Step 3 have to be extracted and refined until they offer an appropriate level of detail for an architecture comparison. For example, if one scenario asks for error management, the appropriate view has to be refined until all components involved with error management become visible.

- **Detect used tactics, styles, and patterns.** The views are analyzed for evidence that supports quality attribute scenarios. The starting point is the suggested list of tactics, styles, and patterns from Step 3.

- **Gather metrics data from available implementations.** Metrics data is gathered from available implementations, if selected in step 3.

- **Verify views with the candidate stakeholders.** The extracted information must be verified and confirmed by the candidate stakeholders.

View and indicator extraction depends on the quality of the architectural documentation with respect to the quality attribute scenarios of step 2. An architectural reconstruction has to be applied if insufficient information is elicited from the documentation or the architects. Architectural reconstruction requires a revisit of the SACAM plan because of the additional activities.

Output:

- verified extracted information
- list of identified indicators, such as tactics, styles, patterns, and metrics data

Dynamics:

View extraction is certainly the most challenging task because of the expected heterogeneity in terminologies, concepts, notations, levels of detail, views, and candidate stakeholders. Note that view extraction and indicator detection are not done sequentially but rather occur at

the same time. Both activities depend on each other: architectural views are of limited value in the SACAM if they do not provide evidence for the support or hindrance of quality attributes.

Example:

We applied an architecture reconstruction of the sunroof. Figure 4 shows the generated Uses View and illustrates the syntax dependencies between processor modules (names that start with "hc11") and other modules that use the processor modules. The extracted tactic is "semantic coherence" of the processor dependencies. In addition, the organization uses the "limited possible variants" tactic by supporting processors of the MC68HC11 family, which is not feasible for a combined sunroof/garage door software product line.

Architecture reconstruction of the sunroof had the following advantages besides the generation of the Uses View:

- The dependencies (relations between the modules in Figure 4) are traceable until source code level.
- Source-fact-extraction tools offer dependency and complexity metrics features.
- The Uses View reflects the as-implemented architecture as opposed to the as-documented architecture.

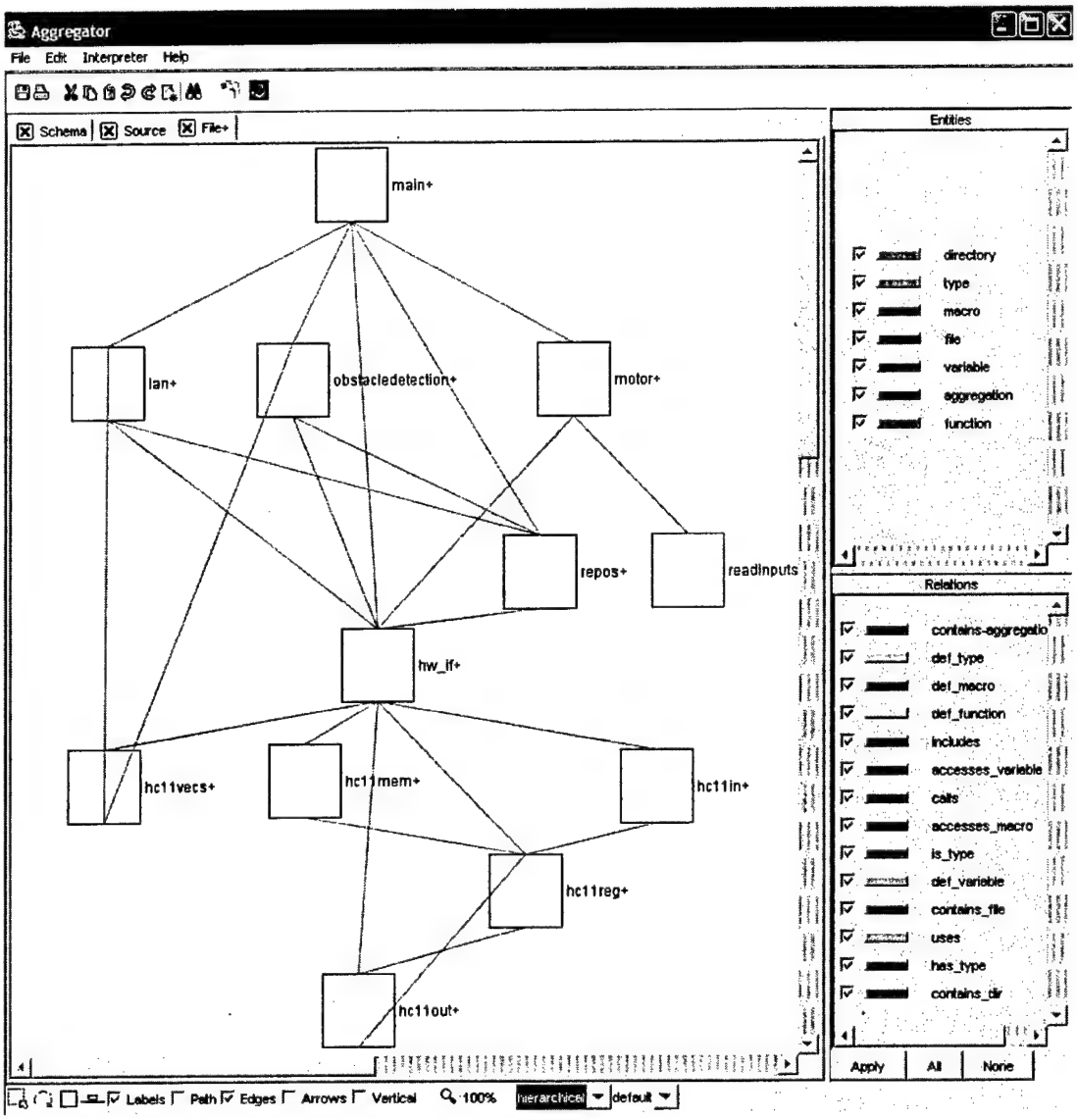


Figure 4: Uses View for the Processor Dependencies of the Sunroof

Step 5 – Scoring

Brief Description: Provides for each architecture candidate a score that indicates how well it supports the quality attribute scenarios generated in the criteria collation workshop of Step 2. Quality analysis models can be used to provide a more formal reasoning about the consequences of the tactics, styles, patterns, and metrics used in the candidate architectures.

Input:

- quality attribute scenarios from Step 2
- verified and extracted architectural views from Step 4

- list of indicators from Step 4
- list of quality attribute models

Activities:

The scoring of the candidates is done for each criterion elicited in the criteria collation workshop of Step 2. Prior to the scoring, each scenario is applied for the candidates.

- **Select the quality attribute models for each quality attribute scenario and calculate the response.** The quality attribute model is fed by the architectural views and the quality attribute scenario. Calculate the response. The quality models for many quality attribute scenarios can be obtained, for example, from the work of Bass and associates [Bass 03].
- **Provide a scoring and reasoning about whether the response of a candidate satisfies the expected response of the quality attribute scenario.** The scores consist of three value ranges: not satisfied, fully satisfied, or satisfied under certain conditions. The conditions provide more feedback to the comparison stakeholders in case particular criteria do not provide enough information or are ambiguous.
- **Collect the quality attribute scenario scores.** This activity collects the individual scores and weights the numbers depending on the scenario's importance.
- **Analyze the scores.** With this activity, the strengths and weaknesses of each candidate in relation to the quality attribute scenarios are identified.

Output:

- scores and reasoning for each candidate

Dynamics:

The scoring is usually done with the comparison and candidate stakeholders. If possible, the architects of each candidate should attend.

Example:

Table 3 illustrates the scores in this step for the sunroof and garage door systems. Each scenario was scored in Step 5 with a number between 0 (architecture does not fit) and 10 (architecture fits perfectly). Each scenario has a weight that is applied to the score. The total scores for scenarios are low because the candidates have different strengths and weaknesses. For example, standards for obstacle detection in the automotive industry are established, whereas comparable laws for garage door openers do not exist in most countries. The bottom row provides the weighted total score for a candidate. Table 3 suggests favoring none of the candidate architectures.

Table 3: Example Scores with Weighted Importance

Quality Attribute Scenario	Weight	Garage Door	Sunroof
Scenario 1 (Obstacle)	1	4/4	9/9
Scenario 2 (Variability)	1	7/7	3/3
Scenario 3 (Diagnosis)	0.8	5/4	5/4
Total	---	15	16

Step 6 – Summary

Brief Description: Provides a summary of the results, along with specific recommendations

Input:

- quality attribute scenarios from Step 2
- extracted architectural views from Step 4
- scoring and reasoning from Step 5

Activities:

- Present results. The results are outlined in a presentation. Final remarks can be considered.
- Write a summary report. The results of the SACAM are summarized in a report that provides feedback and recommendations to the organization.

Output:

- presentation
- summary report with recommendations

Dynamics:

The presentation can be done, after a short preparation, at the end of the scoring workshop.

Example:

The scores from Table 3 favor none of the candidates, so no particular architecture is recommended as the base for the envisioned system. There are three options at this point:

1. Ignore the SACAM results, select a system that implements a particular strategy (for example, the probable disaster strategy "most features"), and extend it.
2. Design the new sunroof/garage door system by performing an architecture design for product lines. To lower the product line adoption barrier, preserve algorithmic parts, such as the obstacle detection in the sunroof.
3. Decide not to proceed with a common product line for sunroofs and garage doors.

Regardless of the final decision, the artifacts generated with the SACAM (views, scenarios, tactics, and patterns) will provide valuable information for subsequent developments with the systems.

6 Conclusions and Future Work

The SACAM provides a rationale for an architecture selection process by comparing the fitness of architecture candidates for an envisioned system based on selected criteria. The method combines an architecture artifact elicitation process with an analysis framework for the selected criteria derived from business goals. The criteria incorporate quality attributes that are refined into scenarios. The SACAM team and stakeholders explore the scenarios for the candidate architectures and try to understand and score the responses given by quality attribute models. The summary report provides architecture recommendations.

It is our belief that using the SACAM yields three major benefits. First, it offers organizations a qualitative approach—beyond quantitative approaches, such as cost—for making selections among alternative architectures. Intuition is replaced by an analysis framework that offers, for example, qualitative reasoning to acquiring organizations. Second, the criteria approach is goal oriented. For example, architectural commonalities and differences are condensed and interpreted with regard to the criteria qualities. Third, comparison on a manageable architectural level offers the ability to evaluate changing business goals in the requirements space with solutions in the design space.

This report provides the first version of the SACAM. The method has potential for other situations that are not investigated here. As development of the method continues, it will be applied to other architecture selection processes.

References

All URLs are valid as of the publication date of this report.

[Arora 95] Arora, V.; Kalaichelvan, K.; Goel, N.; & Munikoti, R. "Measuring High-Level Design Complexity of Real-Time Object-Oriented Systems." *Proceedings of the Annual Oregon Workshop on Software Metrics.* Silver Falls, OR, June 5-7, 1995. Beaverton, Oregon: Oregon Center for Advanced Technology Education, 1995.

[Bachmann 03] Bachmann, F.; Bass, L.; & Klein, M. *Deriving Architectural Tactics: A Step Toward Methodical Architectural Design* (CMU/SEI-2003-TR-004, ADA413644). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr004.html>.

[Barbacci 03] Barbacci, M.; Ellison, R.; Lattanze, A.; Stafford, J.; Weinstock, C.; & Wood, W. *Quality Attribute Workshops (QAWs), Third Edition* (CMU/SEI-2003-TR-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr016.html>.

[Basili 92] Basili, V. R. *Software Modeling and Measurement: The Goal/Question/Metric Paradigm* (CS-TR-2956). College Park, MD: Department of Computer Science, University of Maryland, September, 1992.

[Bass 03] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*, Second Edition. Reading MA: Addison-Wesley, 2003.

[Buschmann 96] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; & Stal, M. *Pattern-Oriented Software Architecture: A System of Patterns.* New York, NY: John Wiley & Sons, 1996.

[CAN 03] CAN in Automation (CiA). *Controller Area Network (CAN), An Overview.* <http://www.can-cia.org/can> (2003).

[Chung 00] Chung, L.; Nixon, B. A.; Yu, E.; & Mylopoulos J. *Non-Functional Requirements in Software Engineering.* Boston, MA: Kluwer Academic Publishers, 2000.

[Clements 02a] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. *Documenting Software Architectures: Views and Beyond.* Reading, MA: Addison-Wesley, 2002.

[Clements 02b] Clements, P.; Kazman, R.; & Klein, M. *Evaluating Software Architectures.* Reading, MA: Addison-Wesley, 2002.

[Faust 03] Faust, D. & Verhoef, C. *Software Product Line Migration and Deployment.* <http://www.cs.vu.nl/~x/pl/pl.html> (2003).

[FlexRay 03] FlexRay Consortium. <http://www.flexray-group.com> (2003).

[Gamma 95] Gamma, E.; Helms, R.; Johnson, R.; & Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, MA: Addison-Wesley, 1995.

[Imagix 03] Imagix Corporation. <http://www.imagix.com> (2003).

[Kazman 97] Kazman, R. & Carrière S. *Playing Detective: Reconstructing Software Architecture from Available Evidence* (CMU/SEI-97-TR-010, ADA330928). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. <http://www.sei.cmu.edu/publications/documents/97.reports/97tr010/97tr010abstract.html>.

[Klein 99] Klein, M. & Kazman, R. *Attribute-Based Architectural Styles* (CMU-SEI-99-TR-022, ADA371802). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. <http://www.sei.cmu.edu/publications/documents/99.reports/99tr022/99tr022abstract.html>.

[Krikhaar 99] Krikhaar, R. L. "Software Architecture Reconstruction" PhD diss., University of Amsterdam, 1999.

[McCabe 70] McCabe, T. J. "A Complexity Measure" *IEEE Transactions on Software Engineering SE-12*, 3 (March 1970):308-320.

[Noble 01] Noble, J. & Weir, C. *Small Memory Software: Patterns for Systems with Limited Memory*. Reading, MA: Addison-Wesley, 2001.

[Stoermer 03] Stoermer, C.; O'Brien, L.; & Verhoef, C. "Moving Towards Quality-Attribute-Driven Software Architecture Reconstruction." *Proceedings of the 10th Working Conference on Reverse Engineering (WCRE)* Victoria, Canada, November 13-16, 2003. Los Alamitos, CA: IEEE Computer Society Press, 2003.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	December 2003	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
SACAM: The Software Architecture Comparison Analysis Method	F19628-00-C-0003

6. AUTHOR(S)

Christoph Stoermer, Felix Bachmann, Chris Verhoef

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2003-TR-006

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2003-006

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Comparing software architectures for any nontrivial system is a difficult task. Software architectures are designed with particular requirements and constraints, and are often poorly documented. However, organizations often need to select a software architecture for future development from several candidate architectures. The Software Architecture Comparison Analysis Method (SACAM) was created to provide the rationale for an architecture selection process by comparing the fitness of architecture candidates for required systems. The SACAM compares architectures based on a set of criteria derived from the business goals of an organization. SACAM was developed in a technical reuse context where an organization investigated architectural commonalities and differences to explore architectural designs for a software product line architecture. This report outlines a first version of the method and its underlying concepts.

14. SUBJECT TERMS	15. NUMBER OF PAGES
software architecture, software architecture selection, software architecture comparison, product line architecture	48

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL